Basic File Attributes

The UNIX file system allows the user to access other files not belonging to them and without infringing on security. A file has a number of attributes (properties) that are stored in the inode. In this chapter, we discuss,

- ls –l to display file attributes (properties)
- Listing of a specific directory
- Ownership and group ownership
- Different file permissions

Listing File Attributes

ls command is used to obtain a list of all filenames in the current directory. The output in UNIX lingo is often referred to as the listing. Sometimes we combine this option with other options for displaying other attributes, or ordering the list in a different sequence. ls look up the file's inode to fetch its attributes. It lists seven attributes of all files in the current directory and they are:

- File type and Permissions
- Links
- Ownership
- Group ownership
- File size
- Last Modification date and time
- File name

The file type and its permissions are associated with each file. Links indicate the number of file names maintained by the system. This does not mean that there are so many copies of the file. File is created by the owner. Every user is attached to a group owner. File size in bytes is displayed. Last modification time is the next field. If you change only the permissions or ownership of the file, the modification time remains unchanged. In the last field, it displays the file name.

For example,

```
$ ls –l
total 72
-rw-r--r--    1 kumar  metal  19514  may  10  13:45    chap01
-rw-r--r--    1 kumar  metal   4174  may  10  15:01    chap02
-rw-rw-rw-    1 kumar  metal     84  feb  12  12:30    dept.lst
-rw-r--r--    1 kumar  metal   9156  mar  12   1999    genie.sh
drwxr-xr-x    2 kumar  metal    512  may   9  10:31    helpdir
drwxr-xr-x    2 kumar  metal    512  may   9  09:57    progs
```

Listing Directory Attributes

ls -d will not list all subdirectories in the current directory
For example,

ls –ld helpdir progs
drwxr-xr-x 2 kumar metal 512 may 9 10:31 helpdir
drwxr-xr-x 2 kumar metal 512 may 9 09:57 progs

Directories are easily identified in the listing by the first character of the first column, which here shows a d. The significance of the attributes of a directory differs a good deal from an ordinary file. To see the attributes of a directory rather than the files contained in it, use ls –ld with the directory name. Note that simply using ls –d will not list all subdirectories in the current directory. Strange though it may seem, ls has no option to list only directories.

File Ownership

When you create a file, you become its owner. Every owner is attached to a group owner. Several users may belong to a single group, but the privileges of the group are set by the owner of the file and not by the group members. When the system administrator creates a user account, he has to assign these parameters to the user:

The user-id (UID) – both its name and numeric representation
The group-id (GID) – both its name and numeric representation

File Permissions

UNIX follows a three-tiered file protection system that determines a file's access rights. It is displayed in the following format:

Filetype owner (rwx) groupowner (rwx) others (rwx)

For Example:

-rwxr-xr-- 1 kumar metal 20500 may 10 19:21 chap02

r w x r - x r - -

owner/user group owner others

The first group has all three permissions. The file is readable, writable and executable by the owner of the file. The second group has a hyphen in the middle slot, which indicates the absence of write permission by the group owner of the file. The third group has the write and execute bits absent. This set of permissions is applicable to others.

You can set different permissions for the three categories of users – owner, group and others. It's important that you understand them because a little learning here can be a dangerous thing. Faulty file permission is a sure recipe for disaster

Changing File Permissions

A file or a directory is created with a default set of permissions, which can be determined by umask. Let us assume that the file permission for the created file is -rw-r--r--. Using **chmod** command, we can change the file permissions and allow the owner to execute his file. The command can be used in two ways:

In a relative manner by specifying the changes to the current permissions
In an absolute manner by specifying the final permissions

Relative Permissions

chmod only changes the permissions specified in the command line and leaves the other permissions unchanged. Its syntax is:

chmod category operation permission filename(s)

chmod takes an expression as its argument which contains:
user category (user, group, others)
operation to be performed (assign or remove a permission)
type of permission (read, write, execute)

Category	**operation**	**permission**
u - user	+ assign	r - read
g - group	- remove	w - write
o - others	= absolute	x - execute
a - all (ugo)		

Let us discuss some examples:

Initially,

-rw-r--r-- 1 kumar metal 1906 sep 23:38 xstart

chmod u+x xstart

-rwxr--r-- 1 kumar metal 1906 sep 23:38 xstart

The command assigns (+) execute (x) permission to the user (u), other permissions remain unchanged.

chmod ugo+x xstart or
chmod a+x xstart or
chmod +x xstart

-rwxr-xr-x 1 kumar metal 1906 sep 23:38 xstart

chmod accepts multiple file names in command line

chmod u+x note note1 note3

Let initially,

-rwxr-xr-x 1 kumar metal 1906 sep 23:38 xstart

chmod go-r xstart

Then, it becomes

-rwx--x--x 1 kumar metal 1906 sep 23:38 xstart

Absolute Permissions

Here, we need not to know the current file permissions. We can set all nine permissions explicitly. A string of three octal digits is used as an expression. The permission can be represented by one octal digit for each category. For each category, we add octal digits. If we represent the permissions of each category by one octal digit, this is how the permission can be represented:

- Read permission – 4 (octal 100)
- Write permission – 2 (octal 010)
- Execute permission – 1 (octal 001)

Octal	Permissions	Significance
0	- - -	no permissions
1	- - x	execute only
2	- w -	write only
3	- w x	write and execute
4	r - -	read only
5	r - x	read and execute
6	r w -	read and write
7	r w x	read, write and execute

We have three categories and three permissions for each category, so three octal digits can describe a file's permissions completely. The most significant digit represents user and the least one represents others. chmod can use this three-digit string as the expression.

Using relative permission, we have,

chmod a+rw xstart

Using absolute permission, we have,

chmod 666 xstart

chmod 644 xstart

chmod 761 xstart

will assign all permissions to the owner, read and write permissions for the group and only execute permission to the others.

777 signify all permissions for all categories, but still we can prevent a file from being deleted. 000 signifies absence of all permissions for all categories, but still we can delete a file. It is the directory permissions that determine whether a file can be deleted or not. Only owner can change the file permissions. User can not change other user's file's permissions. But the system administrator can do anything.

The Security Implications

Let the default permission for the file xstart is

-rw-r--r--

chmod u-rw, go-r xstart or

chmod 000 xstart

This is simply useless but still the user can delete this file
On the other hand,

chmod a+rwx xstart

chmod 777 xstart

-rwxrwxrwx

The UNIX system by default, never allows this situation as you can never have a secure system. Hence, directory permissions also play a very vital role here

We can use chmod Recursively.

chmod -R a+x shell_scripts

This makes all the files and subdirectories found in the shell_scripts directory, executable by all users. When you know the shell meta characters well, you will appreciate that the * doesn't match filenames beginning with a dot. The dot is generally a safer but note that both commands change the permissions of directories also.

Directory Permissions

It is possible that a file cannot be accessed even though it has read permission, and can be removed even when it is write protected. The default permissions of a directory are,

rwxr-xr-x (755)

A directory must never be writable by group and others

Example:

mkdir c_progs

ls –ld c_progs

drwxr-xr-x 2 kumar metal 512 may 9 09:57 c_progs

If a directory has write permission for group and others also, be assured that every user can remove every file in the directory. As a rule, you must not make directories universally writable unless you have definite reasons to do so.

Changing File Ownership

Usually, on BSD and AT&T systems, there are two commands meant to change the ownership of a file or directory. Let kumar be the owner and metal be the group owner. If sharma copies a file of kumar, then sharma will become its owner and he can manipulate the attributes

chown changing file owner and **chgrp** changing group owner

On BSD, only system administrator can use chown
On other systems, only the owner can change both

chown

Changing ownership requires superuser permission, so use **su** command

ls -l note

-rwxr----x 1 kumar metal 347 may 10 20:30 note

chown sharma note; ls -l note

-rwxr----x 1 sharma metal 347 may 10 20:30 note

Once ownership of the file has been given away to sharma, the user file permissions that previously applied to Kumar now apply to sharma. Thus, Kumar can no longer edit *note* since there is no write privilege for group and others. He can not get back the ownership either. But he can copy the file to his own directory, in which case he becomes the owner of the copy.

chgrp

This command changes the file's group owner. No superuser permission is required.

ls –l dept.lst

-rw-r--r-- 1 kumar metal 139 jun 8 16:43 dept.lst

chgrp dba dept.lst; ls –l dept.lst

-rw-r--r-- 1 kumar dba 139 jun 8 16:43 dept.lst

In this chapter we considered two important file attributes – permissions and ownership. After we complete the first round of discussions related to files, we will take up the other file attributes.

- Source: Sumitabha Das, "UNIX – Concepts and Applications", 4th edition, Tata McGraw Hill, 2006